INTERNET-DRAFT
Expires: June 14, 1996
<draft-luotonen-ssl-tunneling-02.txt>

Ari Luotonen
Netscape Communications Corporation
December 14, 1995

Tunneling SSL Through a WWW Proxy

Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as ``work in progress."

To learn the current status of any Internet-Draft, please check the ``1id-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

Table of Contents

Overview

The wide success of SSL (Secure Sockets Layer from Netscape Communications Corporation) has made it vital that the current WWW proxy protocol be extended to allow an SSL client to open a secure tunnel through the proxy.

The HTTPS protocol, which is just HTTP on top of SSL, can be proxied in the same way that currently other protocols are handled by the proxies: to have the proxy initiate a secure session with the remote HTTPS server, and then perform the HTTPS transaction. This is the way

---

FTP and Gopher get handled by proxies, too. However, this approach has two disadvantages:

- The connection between the client and the proxy is normal HTTP, and hence, not secure. This is, however, often acceptable if the clients are in a trusted subnet (behind a firewall).
- The proxy will need to have full SSL implementation incorporated into it.

This document describes a way to do SSL tunneling without an implementation of SSL, in a way that is compatible with the current WWW proxy protocol (as defined by the HTTP/1.0 specification). The existing implementation of SSL tunneling in the Netscape Navigator and the Netscape Proxy Server conform to this specification. A patch implementing this feature also exists for the CERN proxy server (CERN httpd).

General Considerations

When tunneling SSL, the proxy must not have access to the data being transferred in either direction, for sake of security.

The proxy merely knows the source and destination addresses, and possibly, if the proxy supports user authentication, the name of the requesting user.

In other words, there is a handshake between the client and the proxy to establish the connection between the client and the remote server through the proxy. In order to make this extension be backwords compatible, the handshake must be in the same format as HTTP/1.0 requests, so that proxies without support for this feature can still cleanly determine the request as impossible for them to service, and give proper error responses (rather than e.g. get hung on the connection).

SSL tunneling isn't really SSL specific -- it's a general way to have a third party establish a connection between two points, after which bytes are simply copied back and forth by this intermediary.

When SSL tunneling support is put into the SSL source code, it will work for any SSL enhanced application.

The CONNECT Method

The client connects to the proxy, and uses the CONNECT method to specify the hostname and the port number to connect to. The hostname

---

and port number are separated by a colon, and both of them must be specified.

The host:port part is followed by a space and a string specifying the HTTP version number, e.g. HTTP/1.0, and the line terminator (CR LF pair, or a single LF).

After that there is a series of zero or more of HTTP request header lines, followed by an empty line. Each of those header lines is also terminated by the CR LF pair, or a single LF. The empty line is simply another CR LF pair, or another LF.

After the empty line, if the handshake to establish the connection was successful, SSL data transfer can begin.

Example:

```
CONNECT home.netscape.com:443 HTTP/1.0
User-agent: Mozilla/1.1N
```

The advantage of this approach is that this protocol is freely extensible; for example, the proxy authentication can be used.

Example:

```
CONNECT home.netscape.com:443 HTTP/1.0
User-agent: Mozilla/1.1N
Proxy-authorization: basic aGVsbG86d29ybGQ=
```

Proxy Response

After the empty line in the request, the client will wait for a response from the proxy. The proxy will evaluate the request, make sure that it is valid, and that the user is authorized to request such a connection. If everything is in order, the proxy will make a connection to the destination server, and, if successful, send a "200 Connection established" response to the client. Again, the response follows the HTTP/1.0 protocol, so the response line starts with the protocol version specifier, and the response line is followed by zero or more response headers, followed by an empty line. The line separator is CR LF pair, or a single LF.

Example:

HTTP/1.0 200 Connection established
Proxy-agent: Netscape-Proxy/1.1

After the empty line, the proxy will start passing data from the client connection to the remote server connection, and vice versa. At any time, there may be data coming from either connection, and that data should be forwarded to the other connection immediately.

If at any point either one of the peers gets disconnected, any outstanding data that came from that peer will be passed to the other one, and after that also the other connection will be terminated by the proxy. If there is outstanding data to that peer undelivered, that data will be discarded.

Security Considerations

CONNECT is really a lower-level function than the rest of the HTTP methods, kind of an escape mechanism for saying that the proxy should not interfere with the transaction, but merely forward the data. This is because the proxy should not need to know the entire URI that is being accessed (privacy, security), only the information that it explicitly needs (hostname and port number). Due to this fact, the proxy cannot verify that the protocol being spoken is really SSL, and so the proxy configuration should explicitly limit allowed connections to well-known SSL ports (such as 443 for HTTPS, 563 for SNEWS, as assigned by the Internet Assigned Numbers Authority).

Extensibility

The SSL tunneling handshake is freely extensible using the HTTP/1.0 headers; as an example, to enforce authentication for the proxy the proxy will simply use the 407 status code and the Proxy-authenticate response header (as defined by the HTTP/1.0 specification) to ask the client to send authentication information:

HTTP/1.0 407 Proxy authentication required
Proxy-authenticate: ...

The client would then send the authentication information in the Proxy-authorization header:

CONNECT home.netscape.com:443 HTTP/1.0
User-agent: ...
Proxy-authorization: ...

Multiple Proxy Servers

This specification applies also to proxy servers talking to other proxy servers. As an example, double firewalls make this necessary. In this case, the inner proxy is simply considered a client with respect to the outer proxy.

References

[HTTP] T. Berners-Lee, R. Fielding, H. Frystyk, "Hypertext Transfer Protocol -- HTTP/1.0", draft-ietf-http-v10-spec-04.html, October 14, 1995.

[SSL] K. Hickman, T. Elgamal, "The SSL Protocol", draft-hickman-netscape-ssl-01.txt,

Netscape Communications Corporation, June 1995

Author's Address:

Ari Luotonen <ari@netscape.com>
Netscape Communications Corporation
501 E. Middlefield Road
Mountain View, CA 94043
USA